# Exercise 1: Navigation Bar

There is some kind of a navigation bar in almost every Vaadin application. A typical navigation bar is at the top of the page, most of the buttons or links aligned to the left side of the page and one button (for example, "logout"), aligned separately to the right side of the page.

Your task is to create a navigation bar containing four buttons. Three of the buttons should be aligned to the left and one to the right. The caption of the buttons doesn't matter, nor do you need to implement any ClickListeners.

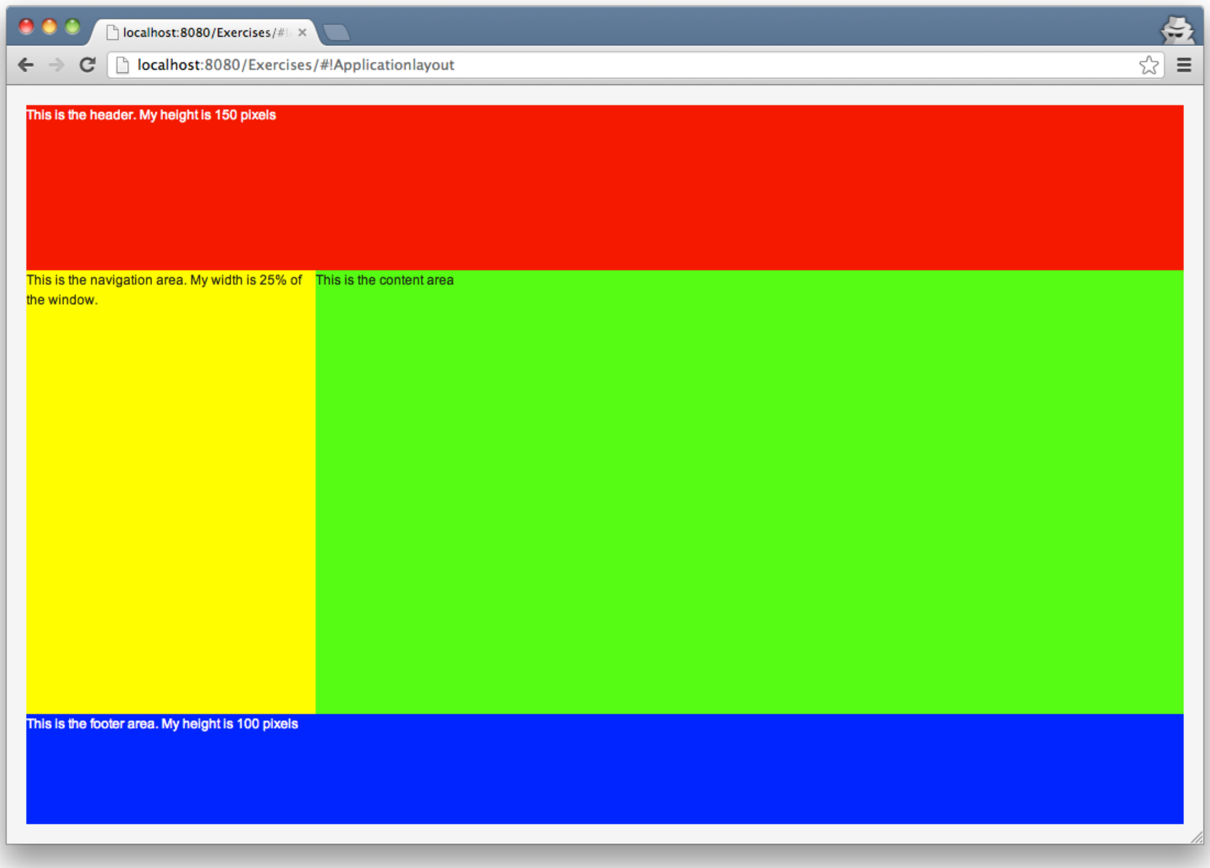
1. Choose an appropriate layout for your navigation bar
2. Create an instance of your layout in `createNavigationLayout()`-method
3. Create four buttons and add them to the layout
4. Align the buttons so that three of the buttons are to the left and one is to the right

Helpful links to Vaadin documentation

- Overview of Layouts
- VerticalLayout and HorizontalLayout
- Layout Formatting
- Composition with CustomComponent

## Exercise 2: Application layout

In this exercise you will learn to create a typical layout structure for a Vaadin application. The goal is to create an application layout containing a header, a footer and a main area that consists of a navigation area and a content area. The screenshot below illustrates the desired result.


localhost:8080/Exercises/#!
localhost:8080/Exercises/#!Applicationlayout

This is the header. My height is 150 pixels

This is the navigation area. My width is 25% of the window.

This is the content area

This is the footer area. My height is 100 pixels

The header marked with a red color should be 100% in width and 150px in height. The footer (in blue) should be 100px in height and 100% in width. The area between the header and the footer should take up the rest of the space in the application. The navigation area (in yellow) should be 25% of the main areas width, while the content area (in green) should be 75% of the content area's width.

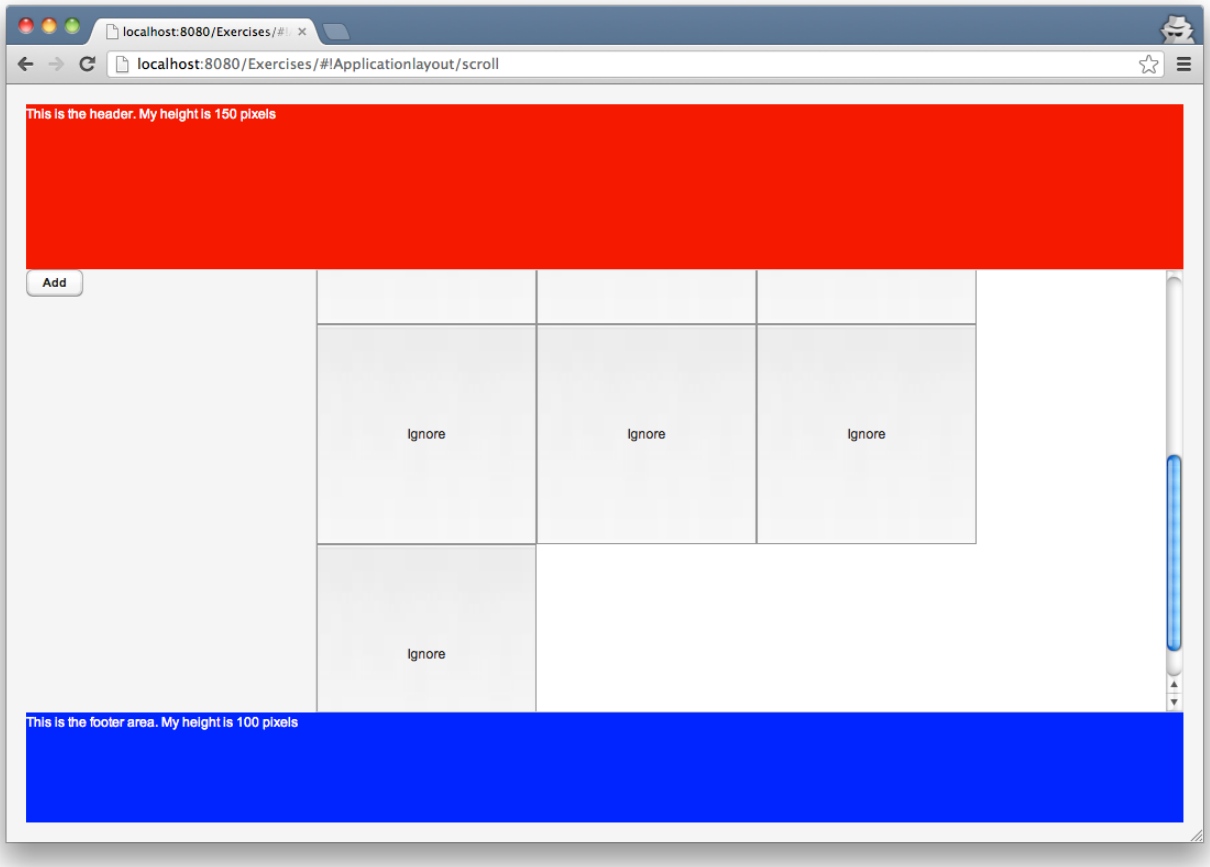
The stub application contains four labels each with a caption and the corresponding color. Your task is to complete the layout. In order to complete the application layout, you need to add one extra layout to the application and adjust the size and expand ratios of the components.

Helpful links to Vaadin documentation

- Overview of Layouts
- Layout Formatting
- Sizing Components

## Exercise 2 continued: Enabling scrollbars

You can continue with this task once you've completed the first part of the application layout. This exercise's purpose is to enable scrollbars in the content area. Start by replacing the content label with a component container, that enables scrolling. After this, replace the navigation label with a button. Implement a click listener which will add a new NativeButton to the content area. You can use the `createButton()` helper method for creating the buttons for the content area.


localhost:8080/Exercises/#!Applicationlayout/scroll
This is the header. My height is 150 pixels
Add
Ignore
Ignore
Ignore
Ignore
This is the footer area. My height is 100 pixels

Note that if I resize the window, the amount of buttons shown on one row in the content area should adjust to my browser size so, that at any given time, the maximum amount of buttons are shown on one row.

The challenges of this task is to figure out how to enable scrollbars and how to make the buttons wrap according to the browser size.

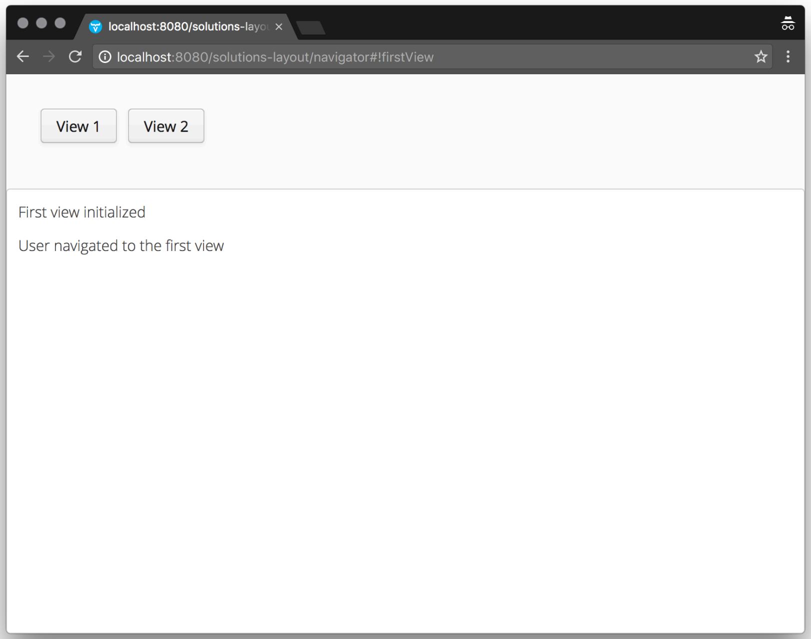
## Exercise 3: Adding Navigator to your application

In this exercise you are provided with a small Vaadin application that uses the Main Layout concept discussed in the previous lecture. The application consists of an UI class called `NavigatorUI`, a main layout class called `MainLayout`, and two view classes. The UI and the view classes are fully functional so you do not need to modify those at all.

Your task is to take the Navigator into use in this example application. Please work with the MainLayout class, where you will find some of the application layout structure already built - including a navigation bar with two buttons and a main content container.

1. First you need to create a Navigator instance. Note that the Navigator must be tied to the current UI instance as well as the view rendering target.
2. Once you have the Navigator instance, you need to register the views. There is a separate method `registerViews` for this purpose. Register `FirstView` as a view instance and `SecondView` as a view class.
3. Implement the `navigateTo` method. In this example this method simply forwards the navigation request to the Navigator instance.
4. Final thing in your main layout constructor is to request the Navigator to navigate to the initial view.
5. For the last step, take a look at the `createNavigationLayout` method. You need to add a click listener for each of the navigation buttons. From these listeners you should request the main layout to navigate to the appropriate view.

The views do some logging output about when they are initialized and when they are navigated to. Keep switching between the two views and you will soon notice the difference between registering a view as an instance or as a class.


localhost:8080/solutions-layo
localhost:8080/solutions-layout/navigator#!firstView
View 1
View 2

First view initialized

User navigated to the first view

Helpful links to Vaadin documentation

- [Navigating in an Application]()